Problem 1

You are tasked with implementing a program that filters a list of students based on certain criteria.

You are provided with a skeleton file, you are not allowed to modify any other part of code except **filter_students()** function from line 17 in **main1.py**.

```python
import sys

def filter_students(student_list, min_age, min_grade, gender):
    """
    Filter students based on minimum age, minimum grade, and gender.

    Parameters:
    - student_list (list): A list of dictionaries where each dictionary represents a student.
    - min_age (int): The minimum age for filtering (inclusive).
    - min_grade (int): The minimum grade for filtering (inclusive).
    - gender (str): The gender for filtering.

    Returns:
    - list: A list of dictionaries containing only those students who meet the specified criteria.
    """
    filtered_students = []
    # WRITE YOUR CODE HERE


    # DONT WRITE ANYTHING BELOW THIS
    return filtered_students

def main():
    # Initialize a list of dictionaries for testing
    student_list = [
        {"name": "John", "age": 20, "grade": 85, "gender": "Male"},
```

Requirements

1. **Input**
   - The skeleton file contains an initialized list of 5 students where each student is represented as a dictionary.
   - The value is passed to the function as follows
     1. student_list (list): A list of dictionaries where each dictionary represents a student.
     2. min_age (int): The minimum age for filtering (inclusive).
     3. min_grade (int): The minimum grade for filtering (inclusive).
     4. gender (str): The gender for filtering.

2. **Filter Students**

- Implement a function called filter_students that takes the following parameters:
    1. student_list: A list of dictionaries where each dictionary represents a student.
    2. min_age: The minimum age for filtering (inclusive).
    3. min_grade: The minimum grade for filtering (inclusive).
    4. gender: The gender for filtering.

- The function should return a new list containing only those students who meet the specified criteria:
    1. Age greater than or equal to min_age
    2. Grade greater than or equal to min_grade.
    3. Gender matching the specified gender.

4. **Output**

- The program should return the list of filtered students.

5. **Save the File Offline**

- Click the “Download Code” button/icon to save the file offline.

6. **Submission**

- Include your Student ID as comment at the top of your code i.e #012345. Rename the main1.py file to q1.py.

7. **Grading**

- (1 point) Correct submission of working code with no errors and loop implementation.
- (5 point) Correct solution to the problem.

Problem 2

You are tasked with implementing a program that finds the Fibonacci number closest to a given number without using any built-in functions for calculating Fibonacci numbers or any mathematical formula.The closest number should be also less than the target number.

You are provided with a skeleton file, you are not allowed to modify any other part of code except **closest_fibonacci()** function from line 12 in **main2.py**.

```python
def closest_fibonacci(target):
    """
    Find the Fibonacci number closest to the given target number that is smaller than the target number,
    using recursion.

    Parameters:
    - target (int): The target number.

    Returns:
    - int: The Fibonacci number closest to the target that is smaller than the target.
    """
    # WRITE YOUR CODE BELOW


    # DONT WRITE ANYTHING BELOW THIS
    pass


def main():

    input_list = [10, 30]

    # Display the output
    for target_number in input_list:
        closest_fib = closest_fibonacci(target_number)
        print(f"\nTarget Number: {target_number}")
        print(f"Closest Fibonacci Number (Smaller than target): {closest_fib}")

```

Requirements

1. **Input**
   - The skeleton file contains an initialized list of 2 numbers.

2. **Closest Fibonacci Calculation**
   - Implement a function called closest_fibonacci that takes a target number as input and recursively calculates Fibonacci numbers until it finds the Fibonacci number closest to the target number.the closest number should be less than target number.
   - Example: Closest Fibonacci number to 10 is: 8

3. **Output**
   - The program should only return the Fibonacci number closest to the target number. No other output or print statements are allowed.

.

4. **Save the File Offline**
   - Click the “Download Code” button/icon to save the file offline.

5. **Submission**
   - Include your Student ID as comment at the top of your code i.e #012345. Rename the main3.py file to q2.py.

6. **Grading**

- (1 point) Correct submission of working code with no errors and loop implementation.
- (5 point) Correct solution to the problem.

Problem 3

You are tasked with implementing a two function called sort_books_by_pages that sorts a list of dictionaries representing books by the number of pages in ascending order without using any built-in sorting functions and search_book_by_title that searches for a specific book by title in the list of dictionaries representing books without using any built-in string functions.Use any algorithm for searching and sorting.

You are provided with a skeleton file, you are not allowed to modify any other part of code except **sort_books_by_pages()** function and **search_book_by_title() function** in **main3.py**.

```
def sort_books_by_pages(books):
    """
    Sorts the list of dictionaries representing books by the number of pages in ascending order without using inbuilt function.
    use any sorting algorithm

    Parameters:
    - books (list): The list of dictionaries representing books.


    Output:
    -Print the output Here itself .Dont Return any value
    """
    # WRITE TOUR CODE BELOW


def search_book_by_title(books, title):

    """
    Searches for a specific book by title in the list of dictionaries representing books without using inbuilt function.
    use any searching algorithm
    Parameters:
    - books (list): The list of dictionaries representing books.
    - title (str): The title of the book to search for.


    Output:
    -Print the output Here itself .Dont Return any value
    """
    # WRITE TOUR CODE BELOW

```

Requirements

1. **Input**

- The skeleton file contains an initialized list of dictionaries representing books.

2. **Sorting Function**

- Implement a function called sort_books_by_pages that takes a list of dictionaries representing books as input and sorts the list by the number of pages in ascending order. You should not use any built-in sorting functions.
- You can use the Selection or Insertion sort algorithm.

3. **Searching Function**
   - Implement a function called search_book_by_title that takes a list of dictionaries representing books and a title as input, and searches for a specific book by title within the list. You should not use any built-in string functions for this search.
   - You can use the Linear or Binary search algorithm.

4. **Output**

- The program should only print the sorted list of books and the search result for a specific title, which are already defined in the provided skeleton file. Do not include any print statements in the code except those already in the skeleton file.

5. **Save the File Offline**
   - Click the “Download Code” button/icon to save the file offline.

6. **Submission**
   - Include your Student ID as comment at the top of your code i.e #012345. Rename the main3.py file to q3.py.

7. **Grading**
   - (1 point) Correct submission of working code with no errors and loop implementation.
   - (5 point) Correct solution to the problem.

Problem 4

You are tasked with implementing a program that allows a user to add student details as input to a function and stores the values in a class Student as object.The class should consist of mainly 4 methods: an initialisation function ,calculate_total_marks ,calculate_percentage and display_student_details.

You are provided with a skeleton file, you are not allowed to modify any other part of code except **create_and_save_student()** function from line 2 in **main4.py**.

```python
def create_and_save_student(name, age, gender, marks):
    # WRITE YOUR CODE HERE
    """
    Function to create a Student object and save details.

    Parameters:
    - name (str): The name of the student.
    - age (int): The age of the student.
    - gender (str): The gender of the student.
    - marks (dict): A dictionary containing subject names as keys and corresponding marks as values.

    Returns:
    - Student: The Student object with the provided details.
    """

    # Student class definition
    """
    Create a claass named Student with the folloing methods as provided below
    """

    # __init__ method definition
    """
    Initializes a new student record with the provided attributes.

    Parameters:
    - name (str): The name of the student.
    - age (int): The age of the student.
    - gender (str): The gender of the student.
    - marks (dict): A dictionary containing subject names as keys and corresponding marks as values.
    """



    # calculate_total_marks method definition
    """
    Calculates the total marks obtained by the student across all subjects.

    Returns:
```

Requirements

1. **Input**

- The skeleton file contains an initialized dictionary containing details of 2 students.
- The value is passed to the function as follows
    1. name (str): The name of the student.
    2. age (int): The age of the student.
    3. gender (str): The gender of the student.
    4. marks (dict): A dictionary containing subject names as keys and corresponding marks as values.

2. **Class Student**

- Implement a class called Student within the create_and_save_student() function with the following methods:

1. init: Initializes a new student record with the provided attributes. This method sets up the initial state of the Student object by assigning values to its attributes.Use the self argument for in every function.

2. calculate_total_marks: Calculates the total marks obtained by the student across all subjects. This method computes the sum of marks obtained in each subject to determine the overall academic performance.

3. calculate_total_percentage: Calculates the percentage of marks obtained by the student. This method computes the percentage of marks obtained based on the total marks and the number of subjects.

4. display_student_details: Displays all the details of the student including name, age, gender, marks obtained in each subject, total marks, and percentage. Also show a message “Great work” if total mark greater than 250 otherwise show “Keep Trying”.

4. **Output**
   - The program should return the object of the class student as provided in the code.

5. **Save the File Offline**
   - Click the “Download Code” button/icon to save the file offline.

6. **Submission**
   - Include your Student ID as comment at the top of your code i.e #012345. Rename the main4.py file to q4.py.

7. **Grading**
   - (1 point) Correct submission of working code with no errors and loop implementation.
   - (5 point) Correct solution to the problem.

File Submissions

Place the q1.py, q2.py, q3.py, and q4.py in a folder and compress to a zip file. Submit it to moodle.